4/5

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
21 June 2001 (21.06.2001)

PCT

(10) International Publication Number
WO 01/45089 A1

(51) International Patent Classification⁷: G10L 15/02

(21) International Application Number: PCT/IB00/01883

(22) International Filing Date:
15 December 2000 (15.12.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
99/7703 15 December 1999 (15.12.1999) ZA

(71) Applicant *(for all designated States except US)*: BRIGHT SPARK TECHNOLOGIES (PROPRIETARY) LIMITED [ZA/ZA]: 260 Surrey Avenue, Ferndale, 2194 Randburg (ZA).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: LILJE, Karl, Rudolph [ZA/ZA]; 17 Morris Avenue, Blairgowrie, 2194 Gauteng (ZA). PRETORIUS, Petrus, Jacobus [ZA/ZA]; 40 Mulbarton Drive, Lonehill, 2062 Gauteng (ZA).

(74) Agents: DE VILLIERS, Christopher, Murray et al.: Spoor and Fisher, P.O. Box 41312, 2024 Craighall (ZA).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

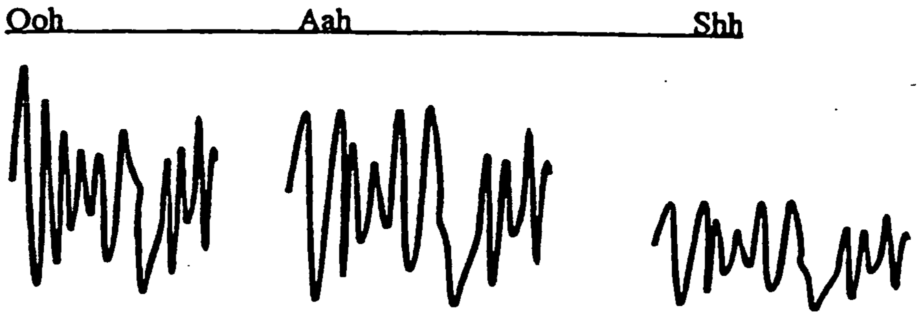
(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *With international search report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

FP03-0314
-00EP-NT
'04. 4.16
SEARCH REPORT

(54) Title: PHONEMES RECOGNITION IN AN AUDIO SIGNAL



(57) Abstract: A method and system for detecting phonemes in an audio signal digitizes an audio signal waveform and detects the number of peaks in a sample of the waveform in order to determine its "roughness". The resulting data is compared with corresponding data of a number of stored audio samples, in order to obtain the best match. A further comparison is then carried out by absolutely subtracting the input data set from the data set corresponding to the matching sample, in order to obtain a final match. The processing required can be carried out rapidly, in real-time or near real-time, and lends itself to live animation applications.

WO 01/45089 A1

PHONEMES RECOGNITION IN AN AUDIO SIGNAL

BACKGROUND OF THE INVENTION

THIS invention relates to a method and system for detecting phonemes in an audio signal.

Various methods have been proposed for the detection of phonemes or other speech components. For example, in voice recognition software, speech is analysed in substantial detail to extract phoneme data from an audio signal, and the resulting phoneme data is arranged into words using various logical rules. Such speech recognition systems are relatively complex and sophisticated, and generally cannot operate in real-time.

It is an object of the invention to provide a relatively efficient phoneme detection method which can operate in real-time or near real-time.

SUMMARY OF THE INVENTION

According to the invention a method of detecting phonemes in an audio signal comprises:

generating an input data set representative of an audio signal waveform;

carrying out an analysis of the input data set to identify amplitude peaks therein;

selecting at least one of a plurality of stored data sets most closely matching the input data set according to said analysis, each stored data set corresponding to a predetermined phoneme; and

comparing said at least one input data set with said at least one selected stored data set according to at least one further criterion, thereby to detect one or more phonemes in the audio signal corresponding to the input data set.

Preferably, a plurality of different selected stored data sets, each corresponding to a predetermined phoneme, are compared with the input data set, the selected stored data set corresponding most closely to the input data set being identified as representing the detected phoneme.

The input data set may be compared with each selected stored data set by subtracting data corresponding to one data set from data corresponding to the other, thereby effectively to determine the closest match between the input data set and the selected stored data sets.

Preferably, amplitude peaks in the input data set are identified by effectively checking the magnitude of one or more points near a given point in the audio signal waveform, recording the result if said one or more points have a lower amplitude than the given point, and repeating the process for additional points in the audio signal waveform.

In the preferred embodiment, the amplitude peaks are identified by incrementing a counter wherever a pair of points equidistant from the given point have a lower amplitude than that of the given point, incrementing the given point, and iterating the process for all points in the audio signal waveform.

The result of the amplitude peak analysis, corresponding to the "roughness" of the audio signal waveform represented by the input data set, may be stored temporarily in a buffer, the contents of which are absolutely subtracted from the contents of respective comparison buffers each containing data corresponding to the "roughness" of a predetermined phoneme.

Preferably, a plurality of data sets are recorded for each predetermined phoneme and data corresponding thereto is stored in respective comparison buffers, the data corresponding to the input data set being subtracted from the data in each respective comparison buffer sequentially in order to enhance the accuracy of detection of the phoneme.

The plurality of data sets may be recorded sequentially by a user.

In a preferred application of the method of the invention, each of the plurality of stored data sets is associated with a predetermined graphic image.

The graphic image typically comprises the face or head of an animated character.

Further according to the invention a system for detecting phonemes in an audio signal comprises:

a digitizer for generating an input data set representative of an audio signal waveform;

a processor for carrying out an analysis of the input data set by identifying amplitude peaks therein;

a memory for storing a plurality of data sets each corresponding to a predetermined phoneme;

a selector for selecting at least one of a plurality of stored data sets most closely matching the input data set according to said analysis, each stored data set corresponding to a predetermined phoneme; and

a comparator for comparing said at least one input data set with said at least one selected stored data set according to at least one further criterion, thereby to detect one or more phonemes in the audio signal corresponding to the input data set.

The comparator is preferably arranged to compare the input data with said at least one selected stored data set by subtracting data corresponding to one data set from data corresponding to the other, thereby effectively to determine the closest match between the input data set and the selected stored data sets.

The processor may be arranged to count amplitude peaks in the audio signal waveform by effectively checking the magnitude of one or more points near a given point in the audio signal waveform, recording the result if said one or more points have a lower amplitude than the given point, and repeating the process for additional points in the audio signal waveform.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a highly simplified block diagram of electronic hardware used to implement the invention;

Figure 2 is a simplified diagram showing the relationship between an input buffer, a property buffer and comparison buffers used by the invention;

Figure 3 is a diagram illustrating the detection of peaks in a sample audio signal;

Figure 4 is a diagram illustrating the comparison process between the contents of the property buffer and a comparison buffer;

Figure 5 is a general process diagram illustrating the operation of the invention in a simplified diagrammatic form;

Figures 6 & 7 are flow diagrams illustrating the classification and comparison processes carried out on the contents of the property buffer in more detail; and

Figures 8 & 9 are examples of waveform diagrams corresponding to the contents of the input buffer and the property buffer, respectively.

DESCRIPTION OF AN EMBODIMENT

The essence of the present invention is a phoneme recognition engine which is designed to operate relatively rapidly and efficiently in order to permit real-time or near real-time phoneme recognition.

The prototype method and system of the invention were developed for use in an animation system which animates a graphical representation of a character in synchronisation with an audio signal, typically a speech signal. In this application, real-time or near real-time processing is important in order to facilitate the synchronisation of the animated graphic with the audio signal.

It will be appreciated that the invention can be applied in other areas, including voice recognition, voice control and other applications, for example.

The hardware used to implement the prototype method and system of the invention is shown in Figure 1.

A microphone 10 is used to record an audio signal from a user or speaker 12. The output of the microphone is fed to a conventional computer sound card 14 which includes an amplifier and a sampling circuit which samples the audio signal at a predetermined sampling rate (eg. 44.1 kHz) and resolution (eg. 16 bits). In the prototype, a Creative Labs Sound Blaster Live^{TM} was used. The digitized audio data is stored in a temporary buffer in RAM of the sound card, which in the prototype system had a size of 512 bytes, corresponding to 11.6ms of sound.

The sampling rate of the audio hardware was set to 22050 samples per second. Thus, the buffer is updated 43.066 times per second (43.066 x 512 = 22050). Each sound sample is represented by an integer in the range –32767 to +32767, and 0 represents silence, or an absence of sound data.

In the graphic animation application for which the prototype system of the invention was developed, the practical buffer size is dependent on the latency expected as a result of the video frame rate. For example, for a frame rate of 30 frames per second, the frames are separated by 33 ms. Therefore, the buffer size divided by the sampling rate should be less than 33 ms.

A sound input interface 16 is able to access the temporary buffer of the sound card and, when sufficient samples have been stored in the temporary buffer (in the present example, 512 samples or 1024 bytes), a "snapshot" of the temporary buffer state is taken by the sound input interface and stored in an Input Buffer 26 (see Figure 2). The contents of the Input Buffer 26 are thus an input data set representative of an audio signal waveform or a portion thereof. Both the sound input interface 16 and the Input Buffer 26 (as well as the Property Buffers and Comparison Buffers described below) are implemented as software.

In the prototype system of the invention, the sound input interface was compliant with the DirectX™ DirectSound™ (version 7) specification of Microsoft Corporation. The sound input interface 16 provides a digital output to a computer 18 in Microsoft Windows Wave file format via the PCI slot on the computer motherboard. The computer has random access memory 20, a hard drive 22 and a monitor or VDU 24. In the prototype system, the computer was an Intel Pentium II with 512 MB of RAM and a 2GB hard drive.

Before the audio sample is processed by the recognition engine proper, certain filtration and preparation steps are carried out. In particular, in order to ensure successful operation of the recognition algorithms, it is desirable to remove noise from the sound data and to normalise the amplitude of the sound sample.

Noise can be introduced into an audio signal by:

- electromagnetic interference from electronic devices in the immediate vicinity.
- Inferior quality cables and connectors used to connect a microphone to the sound capture hardware.
- Radio interference when using a poor quality radio transmitter microphone.
- Handling noise on the microphone during normal operation.
- Ambient sounds in the environment, eg. an audience applauding, or people giving stage instructions.
- Normal breathing from the operator.

If a sound sample falls below 10% of the maximum volume it is set to zero. The maximum volume (amplitude) a sound sample can have is 32767 (positive or negative). Thus if a sound sample falls below 3276 in absolute value, its value is set to 0. The buffer now contains a representation of an audio signal with noise removed.

The next step is to normalise the amplitude of the sound samples. First, the average amplitude (volume) of the sound samples is calculated.

An algorithm iterates through the buffer and finds the average volume (amplitude) of the samples by adding the value of each of the samples and dividing by the total number of samples (512). For example, if the sum of values in the sound buffer is 2500000 then the average volume would be 2500000/512.

If the signal is weak (ie. the buffer has values near 0) the buffer is multiplied by an amount to create sufficient variation from sample to sample for processing purposes.

The audio signal may be weak because of:

- Natural fluctuations in voltage (amplitude) on the audio hardware.
- The cable from the microphone varies in length and thus resistance. The higher the resistance the lower the voltage (amplitude).
- The microphone connected to the audio hardware may be placed in such a way that an unusually low voltage is generated when the user speaks, eg. away from the mouth.
- The microphone connected to the audio hardware may not be manufactured or assembled according to its documented specification.
- A different microphone may be used from the one that is required by the containing system.

The data in the sound buffer is "boosted" by multiplying each sound sample with the highest possible volume (32767) divided by the average volume of the buffer as calculated above.

The buffer is then normalised so that the values fit exactly within the range 0 to 1000. First, the highest volume in the buffer is calculated. An algorithm iterates through the sound data and stores the highest value. Each sample in the buffer is divided by this highest value, resulting in a floating-point value between 0 and 1 (eg. 0.55532). Each sample is multiplied by 1000 so that the value is between 0 and 1000.

The sound buffer now contains a representation of an audio signal with sufficient variation in amplitude to allow a sample-by-sample analysis. Noise from the buffer has been removed. That sound buffer now contains predictable values: the lowest value in the buffer is 0 and the highest value in the buffer is now 1000.

Referring now to Figure 2, the filtered and normalised input data set derived from the audio signal and stored in the input buffer 26 of the sound input interface is now subjected to a classification process as a first step in identifying phonemes in the audio signal. Effectively, the process amounts to a discrete peak count of the audio signal waveform represented by the data in the input buffer, by checking the magnitude of a range of points R around a point P and incrementing a property counter each time a pair of points R equidistant from the point P have a lower magnitude (corresponding to points on the audio signal waveform with a lower amplitude) than that of the point P. The point P is then incremented along the waveform, making this point the new point P, and the process is repeated for the entire buffer.

For example, an Input Buffer's data set may be: [...10 , 20, 30, 40, 20, 10...], and a Property Buffer's data set may be: [0,0,0,0,0,0,0...]

Each element in the Input Buffer is numbered: e63=10, e64=20, e65=30, e66=40 etc. Each element in the Property Buffer is numbered: p1=0, p2=0, p3=0, etc.

We start at element e64. Element e64 is compared to element e63 and to element e65 (i.e. e64-1 and e64+1). If it is higher in magnitude than both then a 1 is added to the Property Buffer at position p1, otherwise the Property Buffer is left untouched (as will happen with the data supplied above).

Element e64 is then compared to element e62 and to element e66 (i.e.e64-2 and e64+2). If it is higher in magnitude than both then a 1 is added to the Property Buffer at position p2 (because it is higher than two of its neighbours). This process is repeated 64 times, the 64th iteration being as follows: Element e64 is compared to element e1 and to element e128 (i.e. e64-64 and e64+64). If it is higher then both then a 1 is added to the Property Buffer at position p64 (because it is higher than 64 of its neighbours). The property buffer now holds, e.g. [1, 1, 0, 0, 0, 0].

Each of the trained buffers (Comparison Buffers) contains a copy of the Property Buffer contents (i.e. the buffer contents resulting from equation 2) at the exact time of training. (for example, while the user was saying 'ooh' the result from equation 2 was stored in a relevant Comparison Buffer, e.g. Ooh number 5).

The Input Buffer contains the raw data from the sound input device. The Property Buffer contains the latest result of equation 2 after it is applied to the Input Buffer. The current Property Buffer is compared to the stored Property Buffers (Comparison Buffers) for each phoneme trained. The Property Buffer is compared by subtracting it from each Comparison Buffer, which yields a numeric result . The lowest result gives an indication to the most likely match.

(e.g.	Comparison Buffer Ooh #5	– Property Buffer	= 72
	Comparison Buffer Aah #2	– Property Buffer	= 172
	Comparison Buffer Shh#6	– Property Buffer	= 511

In this example, Comparison Buffer Ooh#5 has the lowest resulting value and the stored or trained sound represented by it therefore closely matches the sound 'Ooh'. A result of 0 would mean a perfect match).

This process is illustrated diagrammatically in Figure 3 and can be described in programming terms as follows:

The total value of the property counter is stored in a Property Buffer (which is used in the next step of the method).

Let *B(i)* represent the fixed size Input Buffer array of size *BUFFSIZE* where

$B(i) \in [-32727,32727],$ and
$i \in [0, BUFFSIZE].$

Let *P(j)* represent the Property Buffer array of size MAXPROPS where

$P(j) \in [-32727,32727],$
$j \in [1, MAXPROPS],$
$MAXPROPS << BUFFSIZE / 2.(IQ1)$

For calculating *P(j)* the set β_j is defined as follows:

$$\beta_j \subset B(i)\,\alpha \quad iff\{B(i-j) < B(i) > B(i+j), i \in [MAXPROPS, BUFFSIZE - MAXPROPS]\}$$
$.(EQ1)$

(Equation 2)

Figure 3 shows possible locations of B(i), B(i + j) and B(i - j). The range of *i* here shows why the inequality *IQ1* was needed, otherwise points outside the Input Buffer would be accessed.

We can now compute `*P(j)* as follows:

$$P(j) = |\beta_j|,$$
$$j \in [1, MAXPROPS].$$
$$(EQ2)$$

In C++ Computer Code the above translates to:

```
//assuming i, j, P[MAXPROPS] and B[BUFFSIZE] have been instantiated
//where P[] is the Property Buffer and B[] the Input Buffer. (see Fig 7.)

for( j=1 ; j<MAXPROPS ; j++) P[j]=0; //initialise the property values

for( i=MAXPROPS ; i<BUFFSIZE-MAXPROPS ; i++)
{         for(j=1 ; j<MAXPROPS ; j++)
                  if( ( B[i]>B[i+j] )&( B[i]>B[i-j] ) ) P[j]++;
}
```

The above small, yet efficient function allows the invention to perform a property analysis of the Input Buffer contents in a fraction of the time required to capture the data in the buffer. The Property Buffer provides the input for training and comparison methods used by the invention and updates itself every time a new Input Buffer is captured. This method of classification has been found to be more consistent with phonemes than other methods, such as those employing Fourier transforms.

The result of the above process is that the Property Buffer contains the results of a "roughness classification" represented by the number of amplitude peaks in the Input Buffer. This information is now compared with corresponding data for each of a number of Comparison Buffers which contain previously stored data corresponding to different phonemes which the system has been "trained" to recognise. Preferably, two or more sets of data are stored for each phoneme (seven sets in the prototype system) and stored in respective

Comparison Buffers, with each Comparison Buffer being indexed to a respective Property Buffer and further being identified by an alphanumeric character, the use of which will be apparent from the description below.

Effectively, in order to identify a phoneme which is stored in the Input Buffer, the system first applies the above described classification process to populate the Property Buffer with data, and the data of the Property Buffer is then "absolutely subtracted" from the respective Property Buffers in each Comparison Buffer. The resulting totals are compared, with the lowest total corresponding to the Comparison Buffer whose contents most closely resemble those of the Property Buffer in question. This process is described below, with reference to Figure 4:

In Figure 4, the grey region represents the area between the curves *C(k)* and *P(k)*, which is calculated using the following formula:

Let $C_i(k)$ be the *i*th stored Property Buffer in a Indexed Comparison Buffer, and *P(k)* the current Property Buffer, where $k \in [1, MAXPROPS]$ and *i* is the index of that stored Property Buffer. We define *MAXINDEX* as the maximum number of stored Property Buffers in one indexed Comparison Buffer. Then $i \in [1, MAXINDEX]$.

Thus the total area between the *i*th stored Property Buffer in a indexed Comparison Buffer and the current Property Buffer , is T_i ,where:

$$T_i = \sum_{k=1}^{MAXPROPS} |C_i(k) - P(k)|$$

The Total for a phoneme is just the sum of the totals, T_i, for an indexed Comparison Buffer for that phoneme. Each indexed Comparison Buffer represents a phoneme:

$$T = \sum_{i=1}^{MAXINDEX} T\,i$$

Associated with the relevant Comparison Buffer is an alphanumeric character which has previously been associated with the stored phoneme represented by that Comparison Buffer. This character can then be utilised to trigger other events. For example, can be used to trigger sounds, animations or MIDI events.

The overall recognition process of the invention is described in diagrammatic form in Figure 5, and the above described classification and comparison processes are shown diagrammatically in Figures 6 and 7, respectively.

An example of an application for the present invention is in an animation triggering system. The invention is used by this system to distinguish different basic phonemes from each other. The prototype system in question has a library of predrawn picture files containing different mouth shapes which correspond to the appearance of a human face uttering corresponding phonemes.

Before the system can be used, it must be trained for a given user/speaker. Each phoneme required to be stored is recorded repeatedly (seven times in the prototype system) and the data for each phoneme is stored in a respective Comparison Buffer. The software controlling the system associates the alphanumeric character identifying each phoneme with a predetermined graphic image, typically the face or head of an animated graphics character. Character image files are stored as sets of animation frames loaded into the system RAM 20. The software links each set of frames to a virtual button displayed on a monitor, and when the user clicks this button the frames are displayed one at a time on the screen. The user then clicks a button which tells the systems that the user is beginning a performance. When the user speaks, the graphics systems automatically displays the graphic, indexed in real-time by the audio recognition system.

As the user speaks, the sound card loads audio data into its temporary buffer. When the sound card has data ready for processing it sets a flag in the computer's RAM that is monitored by the operating system. The operating system then notifies the calling program of the flag's status. The calling program then calls a function to copy data from the sound card into the computer's RAM and tells the sound card to continue capturing.

The data in the RAM is demarcated as a one-dimensional array of 512 numbers in the range [-32767; 32767] by the calling program. This data is then available to the calling program for interpretation/processing.

The following basic steps are follows:

- User says something
- Sound card captures into internal buffer
- Operating system notifies program
- Program accesses numbers representing the sound, e.g. [1231, 123, 5656, 4456, 4534, 3453, 31534...]

The process represented by Equation 2 above can be explained in greater detail as follows.

First, a Classification Array is created with 64 elements, all of which are set to zero. i.e. [0,0,0,0,0,0,0,0,0,0,...]

The input data set created by the sound card and standard sound capture routine is a one-dimensional array of 512 numbers in the range (-32767; 32767) i.e. [23, 100, 45, 123, 5612, 123, 545, 12312, 4454, 1231, 3234, ...]

Each element in the input data set is compared to the elements that surround it in the array. If the element is higher in value than the elements which are on either side of it the number 1 is added to the Classification Array at position one. If the element is higher than the elements on either side of it plus 1 (i.e. the neighbours' neighbours on either side) then a 1 is added to the Classification Array at position 2. If the element is higher then elements which are 3, 4, 5, ...64 units on either side then the Classification Array is incremented at the position equal to the number of units away minus 1.

This creates a Classification Array that represents a graph of the "roughness" of the wave and the ratio of roughness of a certain order in relation to the roughness present in the wave of another order. Each element in the Classification Array is subtracted from the corresponding element in the Reference Arrays stored while training each phoneme or sound. If the resulting difference is less than or equal to a user definable tolerance value then it is considered effectively identical at that position and a counter is incremented. If no result exists below a tolerance level (eg. if the tolerance were set to 130 and the lowest result were 140) then the system assumes the incoming signal is noise or an unrecognizable sound and takes no action. Each element in the array is subtracted this way. The counter value is stored and labeled with the phoneme currently under comparison. This comparison process is repeated for each phoneme that was trained. The counters for each phoneme are then compared and the counter with the highest value is considered to be the most accurate match and the corresponding label is returned to the calling program, e.g. 'A'.

An example of the output:

The contents of the Input Buffer [23, 123, 12,3 12, 3, 12, 3123, 1231,23 12,3 12, 345, 3,1 3,12 3,123, 1...] corresponding to the spoken sounds 'Ooh' or 'Aah', or 'Shh' (see Figure 8) are processed by Equation 2, which results in a Classification Array [1, 3, 4, 7, 6, 3, 2...] (see figure 9).

The result, a single alphanumeric character (e.g. 'A') is flagged as the current sound being said by the user. The images stored in system RAM each have a label, e.g. 'A', which corresponds to the result given by the audio analysis part of the system.

More specifically, consider a more specific example:
C=A Trained Comparison Buffer contents currently being examined, e.g. Oooh #5
P=The current Property Buffer contents obtained by passing the Input Buffer contents through Equation 2

There are 64 elements in the Property Buffer. For clarity only 2 have been illustrated here:

C[3] has a value of 1124, C[4] = 991
P[3] has a value of 1029, P[4] = 1022

C[3] – C[4] = 133
C[4] – P[4] = 7

133 + 7 = 140

140 is the result for this comparison.

The same process happens with Aah #2 (and all the others) but Aah #2 ultimately results in the number 400, therefore the current sound matches the Ooh sound better than the Aah sound.

The graphics part of the system operates on a loop and continually draws images on the screen, depending on the result from the audio analysis part, as well as from conventional user input (e.g. the user presses the "Look Left" button and says "Ooh" to create an animated graphic of a head turning to the left and moving its mouth in synchronisation with the sound "Ooh".)

An example:

- The user presses the "Look Left" button while saying "Ooh".
- The user interface detects the user clicking a button and tells the program to change the head pictures to those of the character looking left.
- The audio part of the system recognizes the "Ooh" sound and tells the graphics part to prepare to draw an image that shows the character saying "Ooh".

The drawing procedure is as follows:
The background image data is copied into the video card's RAM, however this data is copied to a location where it cannot be seen yet. The body of the character is drawn over the background, ignoring all colours that have no value (i.e. a Red Green Blue value of 0,0,0 ... which looks like black). The correct head is drawn over the body, showing the mouth in the detected shape. The eyes are drawn over the head. The unseen image is then copied to the video card's RAM, in a location that is visible to the user/audience. (When you look at a computer monitor you are effectively looking directly at the video card's RAM as seen by the monitor.)

This procedure repeats until the user selects another option with the mouse or keyboard or says a different sound.

Because the entire procedure happens in just 40 milliseconds (25 times a second) an illusion of continuous motion is created.

CLAIMS

1. A method of detecting phonemes in an audio signal comprising:

   generating an input data set representative of an audio signal waveform;

   carrying out an analysis of the input data set to identify amplitude peaks therein;

   selecting at least one of a plurality of stored data sets most closely matching the input data set according to said analysis, each stored data set corresponding to a predetermined phoneme; and

   comparing said at least one input data set with said at least one selected stored data set according to at least one further criterion, thereby to detect one or more phonemes in the audio signal corresponding to the input data set.

2. A method according to claim 1 wherein a plurality of different selected stored data sets, each corresponding to a predetermined phoneme, are compared with the input data set, the selected stored data set corresponding most closely to the input data set being identified as representing the detected phoneme.

3. A method according to claim 2 wherein the input data set is compared with each selected stored data set by subtracting data corresponding to one data set from data corresponding to the other, thereby effectively to determine the closest match between the input data set and the selected stored data sets.

4. A method according to claim 3 wherein amplitude peaks in the input data set are identified by effectively checking the magnitude of one or more points near a given point in the audio signal waveform, recording the result if said one or more points have a lower amplitude than the given point, and repeating the process for additional points in the audio signal waveform.

5. A method according to claim 4 wherein the amplitude peaks are identified by incrementing a counter wherever a pair of points equidistant from the given point have a lower amplitude than that of the given point, incrementing the given point, and iterating the process for all points in the audio signal waveform.

6. A method according to claim 4 or claim 5 wherein the result of the amplitude peak analysis, corresponding to the "roughness" of the audio signal waveform represented by the input data set, is stored temporarily in a buffer, the contents of which are absolutely subtracted from the contents of respective comparison buffers each containing data corresponding to the "roughness" of a predetermined phoneme.

7. A method according to any one of claims 3 to 6 wherein a plurality of data sets are recorded for each predetermined phoneme and data corresponding thereto is stored in respective comparison buffers, the data corresponding to the input data set being subtracted from the data in each respective comparison buffer sequentially in order to enhance the accuracy of detection of the phoneme.

8. A method according to claim 7 wherein the plurality of data sets are recorded sequentially by a user.

9. A method according to any one of claims 1 to 8 wherein each of the plurality of stored data sets is associated with a predetermined graphic image.

10. A method according to claim 9 wherein the graphic image comprises the face or head of an animated character.

11. A system for detecting phonemes in an audio signal comprising:

a digitizer for generating an input data set representative of an audio signal waveform;

a processor for carrying out an analysis of the input data set by identifying amplitude peaks therein;

a memory for storing a plurality of data sets each corresponding to a predetermined phoneme;

a selector for selecting at least one of a plurality of stored data sets most closely matching the input data set according to said analysis, each stored data set corresponding to a predetermined phoneme; and

a comparator for comparing said at least one input data set with said at least one selected stored data set according to at least one further criterion, thereby to detect one or more phonemes in the audio signal corresponding to the input data set.

12. A system according to claim 11 wherein the comparator is arranged to compare the input data with said at least one selected stored data set by subtracting data corresponding to one data set from data

corresponding to the other, thereby effectively to determine the closest match between the input data set and the selected stored data sets.

13. A system according to claim 12 wherein the processor is arranged to identify amplitude peaks in the audio signal waveform by effectively checking the magnitude of one or more points near a given point in the audio signal waveform, recording the result if said one or more points have a lower amplitude than the given point, and repeating the process for additional points in the audio signal waveform.

FIG 1

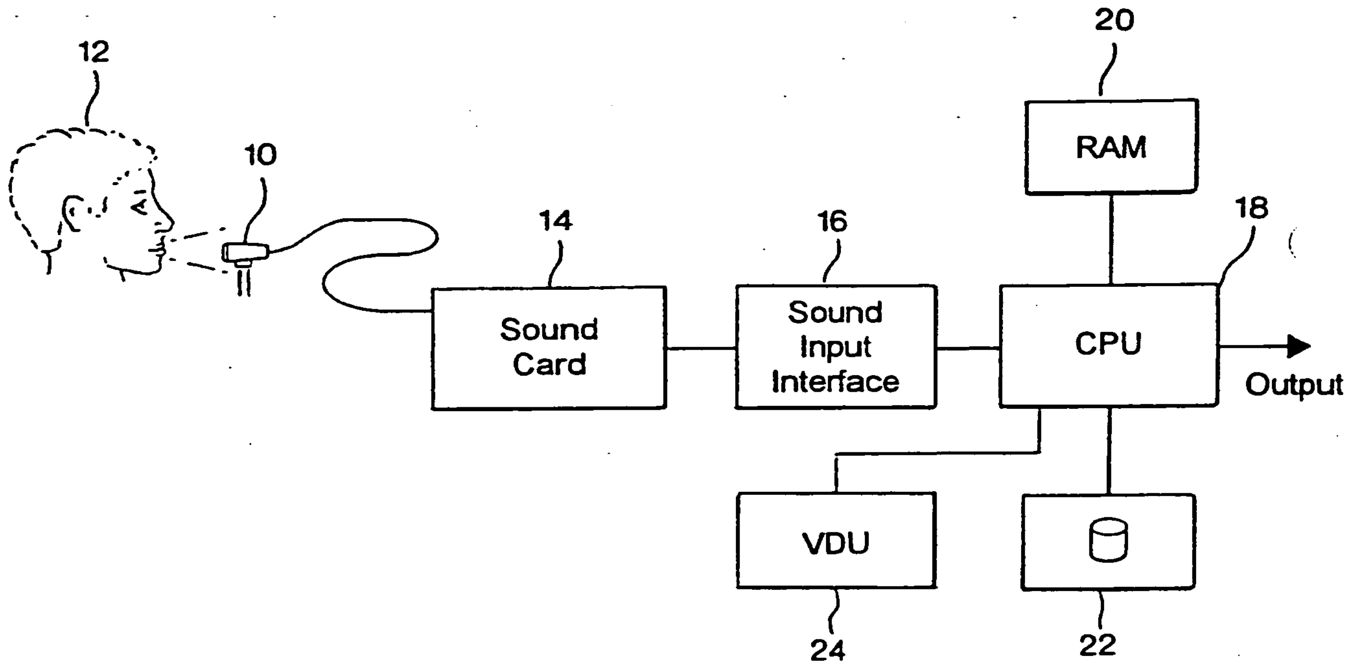

12
10
14
Sound Card
16
Sound Input Interface
20
RAM
18
CPU
Output
VDU
24
22

FIG 2

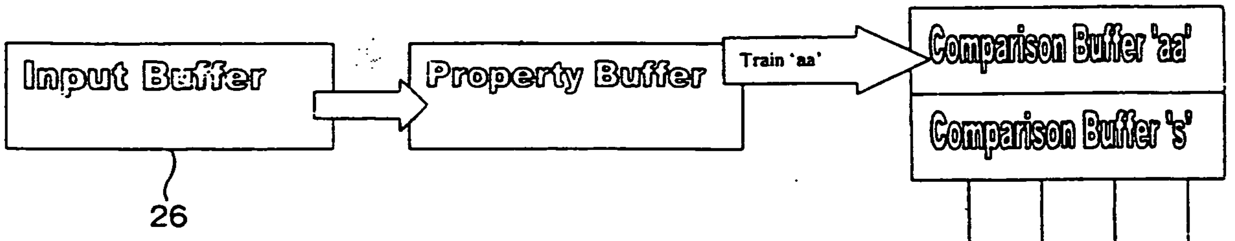

Input Buffer
26
Property Buffer
Train 'aa'
Comparison Buffer 'aa'
Comparison Buffer 's'
Comparison Buffer 'oo'

FIG 3

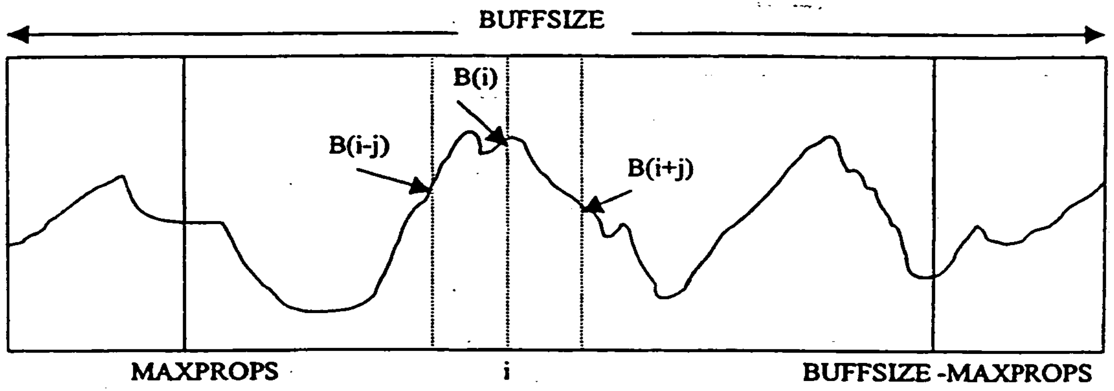


FIG 4

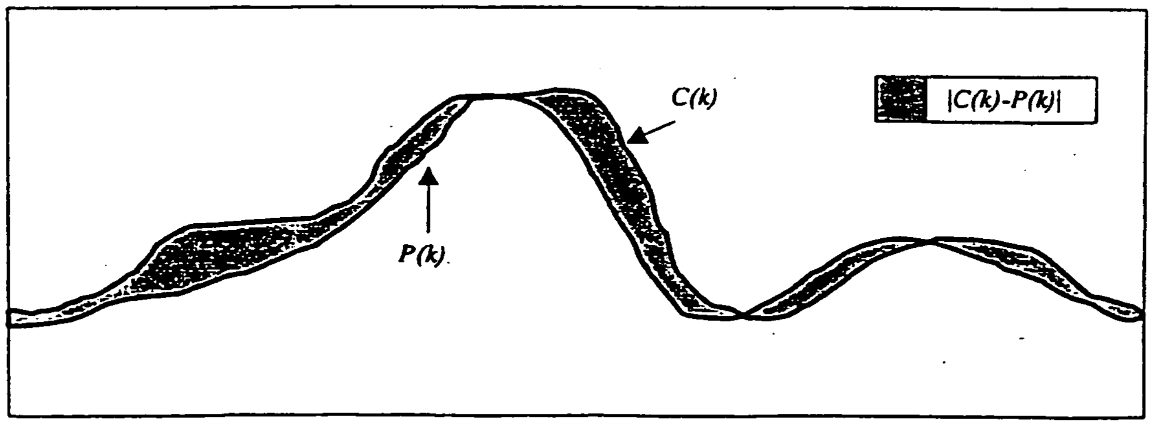
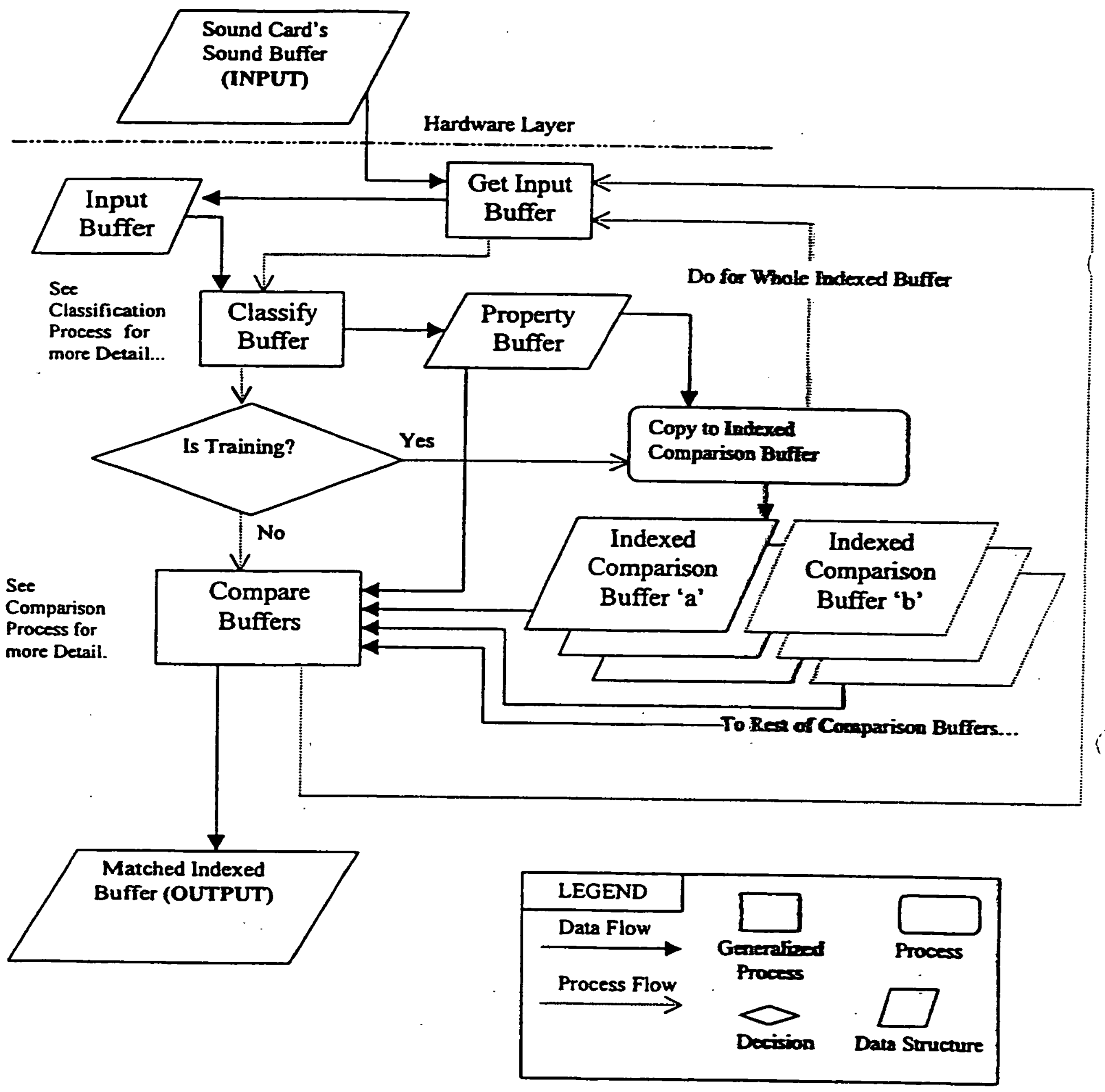


FIG 5


Sound Card's Sound Buffer (INPUT)
Hardware Layer
Get Input Buffer
Input Buffer
Do for Whole Indexed Buffer
See Classification Process for more Detail...
Classify Buffer
Property Buffer
Is Training?
Yes
Copy to Indexed Comparison Buffer
No
Indexed Comparison Buffer 'a'
Indexed Comparison Buffer 'b'
See Comparison Process for more Detail.
Compare Buffers
To Rest of Comparison Buffers...
Matched Indexed Buffer (OUTPUT)
LEGEND
Data Flow
Process Flow
Generalized Process
Process
Decision
Data Structure

Note: The 'Is Training' flag is set asynchronously to this process when a certain event is triggered. E.g. the user clicks an application defined " Train 'a' " Button. A Windows message linked to that button is broadcasted. The application then traps the message, interprets it as a training request and sets the 'Is Training' flag to 'Yes'. It also passes the alphanumeric specifying a particular Indexed Training Buffer that will be trained.

FIG 6

Predefined Values:

MAXPROPS is the size of the Property Buffer.

BUFFSIZE is the size of the Input Buffer.

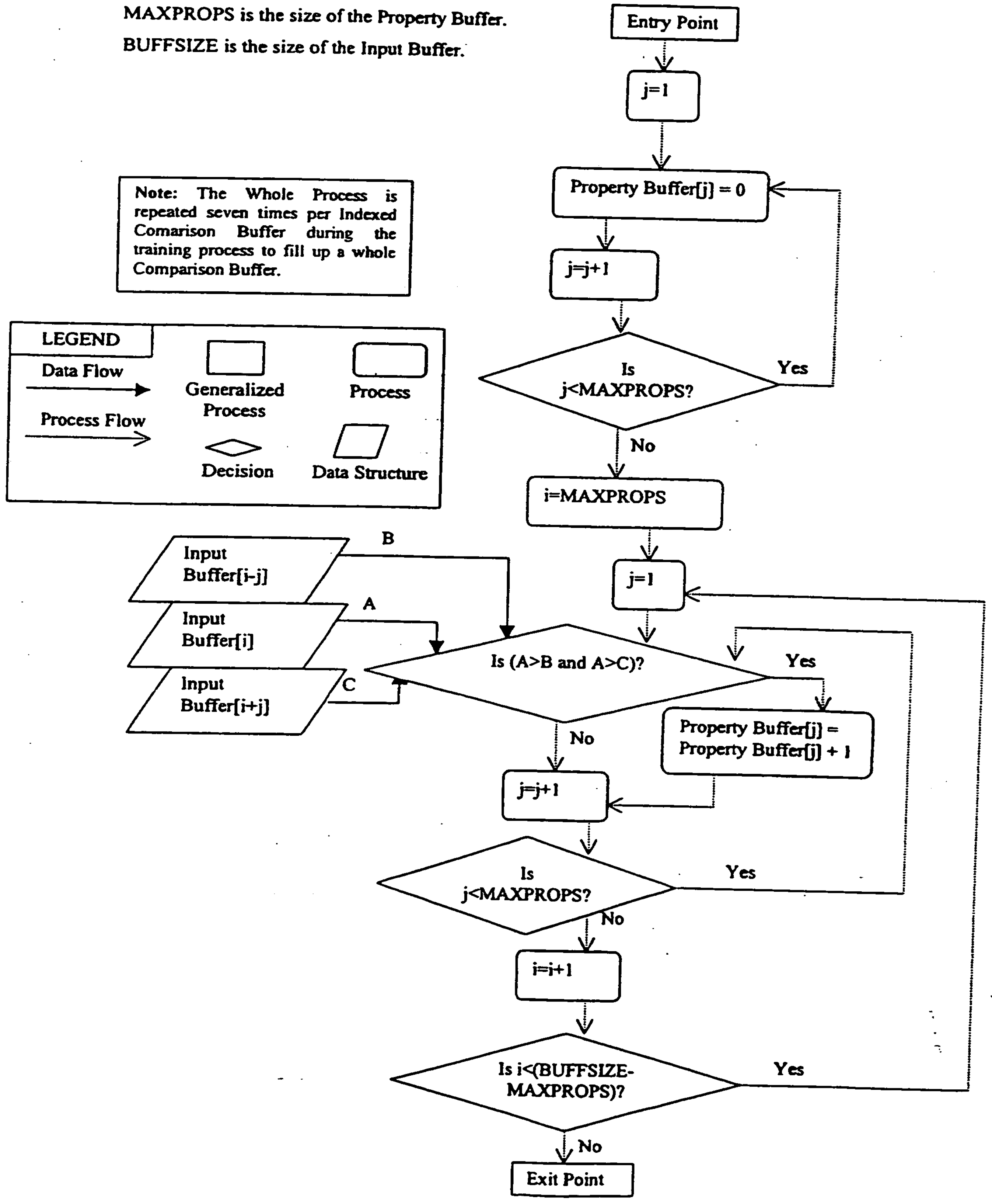
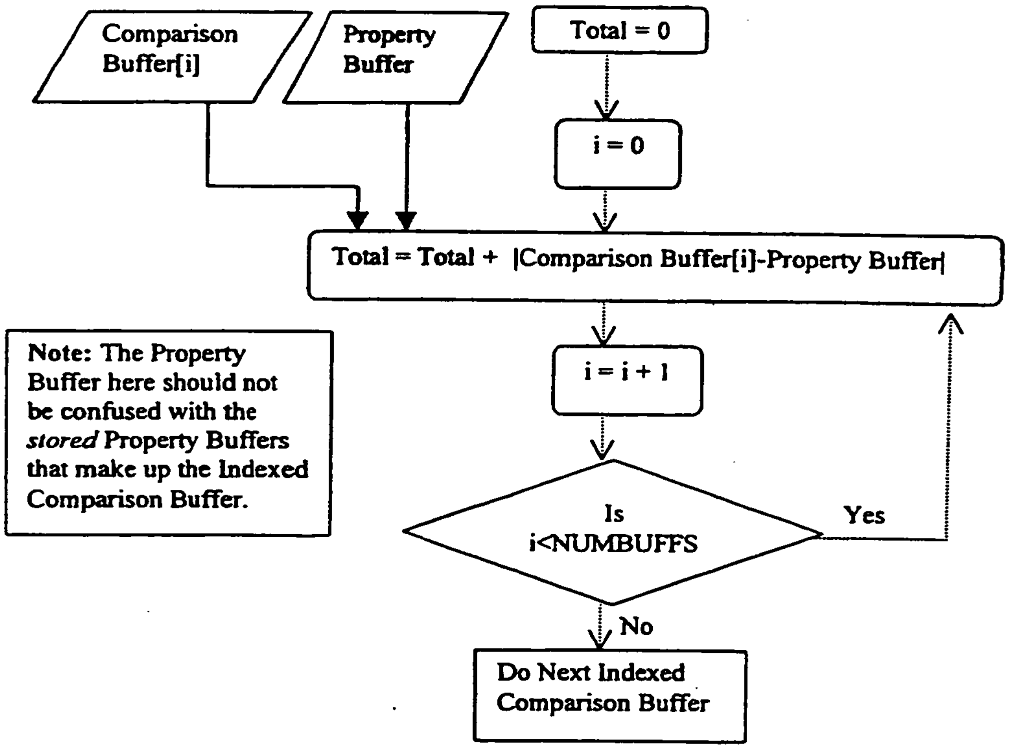


FIG 7

Predefined Values:

NUMBUFFS is the number of stored Property Buffers per Indexed Comparison Buffer (i.e. the Maximum Index which is currently seven).

The Comparison Buffer's index points it to a stored Property Buffer.


Comparison Buffer[i]
Property Buffer
Total = 0
i = 0
Total = Total + |Comparison Buffer[i]-Property Buffer|
Note: The Property Buffer here should not be confused with the *stored* Property Buffers that make up the Indexed Comparison Buffer.
i = i + 1
Is i<NUMBUFFS
Yes
No
Do Next Indexed Comparison Buffer

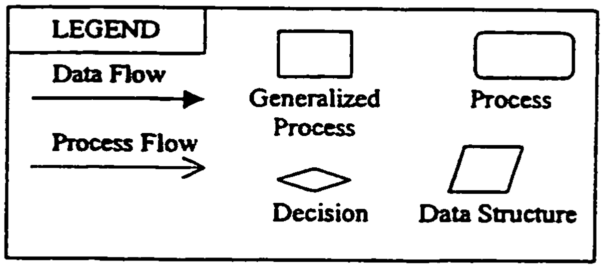


FIG 8

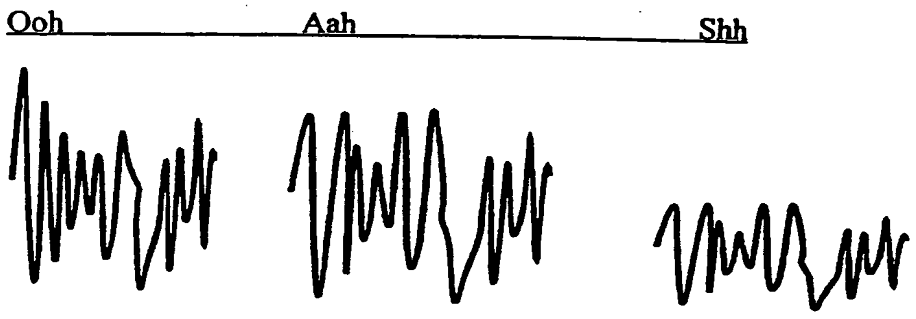
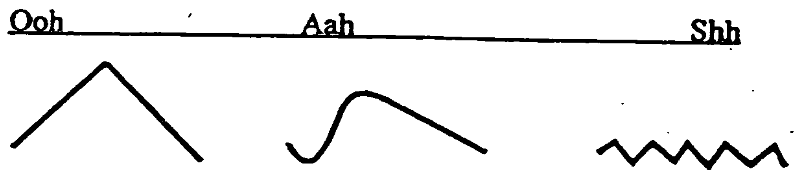


FIG 9



INTERNATIONAL SEARCH REPORT

International Application No
PCT/IB 00/01883

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G10L15/02

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G10L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, WPI Data, INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5 708 759 A (KEMENY) 13 January 1998 (1998-01-13)	1,2,11
Y	column 1, line 13-17	3-10,12, 13
	column 1, line 22-37	
Y	EP 0 768 639 A (ESPAR FIGUERAS ORIOL) 16 April 1997 (1997-04-16) page 4, line 35-39	3-8,12, 13
Y	US 5 313 522 A (SLAGER) 17 May 1994 (1994-05-17) abstract	9,10
A	US 4 624 010 A (TAKEBAYASHI) 18 November 1986 (1986-11-18) the whole document	1-8, 11-13

☐ Further documents are listed in the continuation of box C.

[✓ X] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search 28 February 2001	Date of mailing of the international search report 07/03/2001
Name and mailing address of the ISA European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Authorized officer Quélavoine, R

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

Inter nal Application No
PCT/IB 00/01883

Patent document cited in search report		Publication date	Patent family member(s)			Publication date
US 5708759	A	13-01-1998	NONE			
EP 0768639	A	16-04-1997	ES	2110899	A	16-02-1998
			AU	5400296	A	07-11-1996
			WO	9633486	A	24-10-1996
US 5313522	A	17-05-1994	NONE			
US 4624010	A	18-11-1986	JP	1652145	C	30-03-1992
			JP	3006517	B	30-01-1991
			JP	58130396	A	03-08-1983
			DE	3364573	D	28-08-1986
			EP	0085543	A	10-08-1983